Combining Data and Computation Transformations for Fine-Grain Reconfigurable Architectures

Cristiano B. de Oliveira and Eduardo Marques
Institute of Mathematical and Computer Sciences, University of Sao Paulo, Brazil
{cbacelar,emarques}@icmc.usp.br

Abstract

This project aims to develop techniques for optimal code generation for High Performance Computing using reconfigurable hardware. Some problems faced are related to memory access, since the latency for data reading and writing became a bottleneck for HPC applications. The project is a branch of a system for hardware acceleration in HPC applications currently being developed at University of Sao Paulo, Brazil. This system intends to facilitate the development of HPC architectures for efficiently exploit of parallelism.

Objectives

The project is based on LALP language, aiming to allow that both execution flow and memory access can be controlled by the developer.
The main objectives are:

- **Evaluation of compilation and optimization focused on memory access, both on-chip and off-chip;**
- **Development of techniques to assist the implementation of HPC applications through optimized data management for reconfigurable hardware synthesis;**
- **Implementation of directives for memory mapping of arrays that supports pipelining and non-pipelining access operations;**
- **Improvement of LALP functionalities by adding mechanisms for large massive data handle.**

The current work is focused on translating transformed C source code into LALP code by using ROSE [1, 2] compiler. No results are available yet nevertheless some benchmark tests are expected to be performed soon.

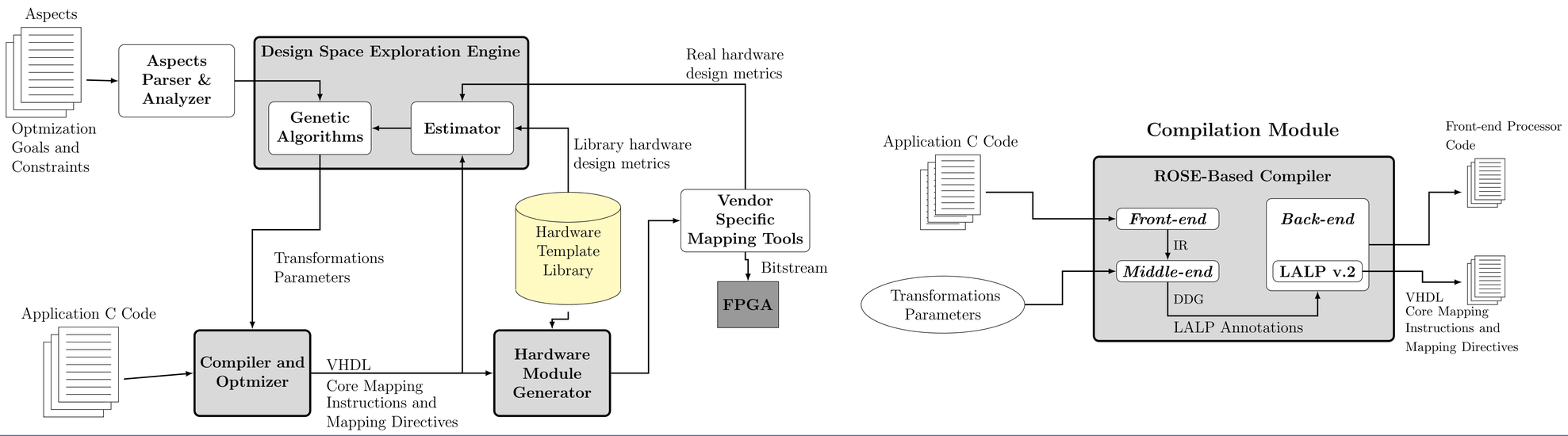
Language for Aggressive Loop Pipelining

LALP [3] (Language for Aggressive Loop Pipelining) is a language focused on mapping loop structures while explores the parallelism of portions of code involving loops. By using LALP is possible to achieve high control of a program execution flow through definition of the number of clock cycles delayed/executed for specifics operations.

```
example(out int xl_final,out int xr_final, out bit
    done, in bit init)
{
  { ... }

  counter (i=0; i<16; i++@19);
  i.clk_en = init;

  p_addr = i;
  p_addr = 16 when i.done@1;
  p_addr = 17 when i.done@3;
  P_ENC.address = p_addr;

  ...

  d = xl & 0xff    when i.step@5;
  c = (xl >>  8) & 0xff when i.step@5;
  b = (xl >> 16) & 0xff when i.step@5;
  a = (xl >> 24) & 0xff when i.step@5;

  s_addr = i;
  s_addr = (0 + a) when i.step@6;
  s_addr = (256 + b) when i.step@7;
  s_addr = (512 + c) when i.step@8;
  s_addr = (768 + d) when i.step@9;

  ...
}
```

Listing 1: Piece of LALP code.

System for Hardware Acceleration in HPC Applications



Acknowledgements

The authors would like to thank the Institute for Mathematical and Computer Sciences (ICMC) at the University of Sao Paulo (USP), for the infrastructure provided to develop this work and FAPESP (the Foundation for Support Research of the State of Sao Paulo) for the financial support provided. We also would like to special thank professor Pedro Diniz (USC-ISI) for his visit to ICMC in May/2011 and his contributions for the definition of this research.

References

[1] **D. Quinlan and C. Liao, "The ROSE Source-to-Source Compiler Infrastructure," in *Cetus Users and Compiler Infrastructure Workshop, in conjunction with PACT 2011*, 2011, pp. 1–3.**

[2] **C. Liao, D. Quinlan, T. Panas, and B. de Supinski, "A ROSE-based OpenMP 3.0 research compiler supporting multiple runtime libraries," *Beyond Loop Level Parallelism in OpenMP: Accelerators, Tasking and More*, pp. 15–28, 2010.**

[3] **R. Menotti, J. M. P. Cardoso, M. Fernandes, and E. Marques, "LALP: A Language to Program Custom FPGA-Based Acceleration Engines," *International Journal of Parallel Programming*, vol. 40, pp. 262–289, 2012.**